Kernel Optimization

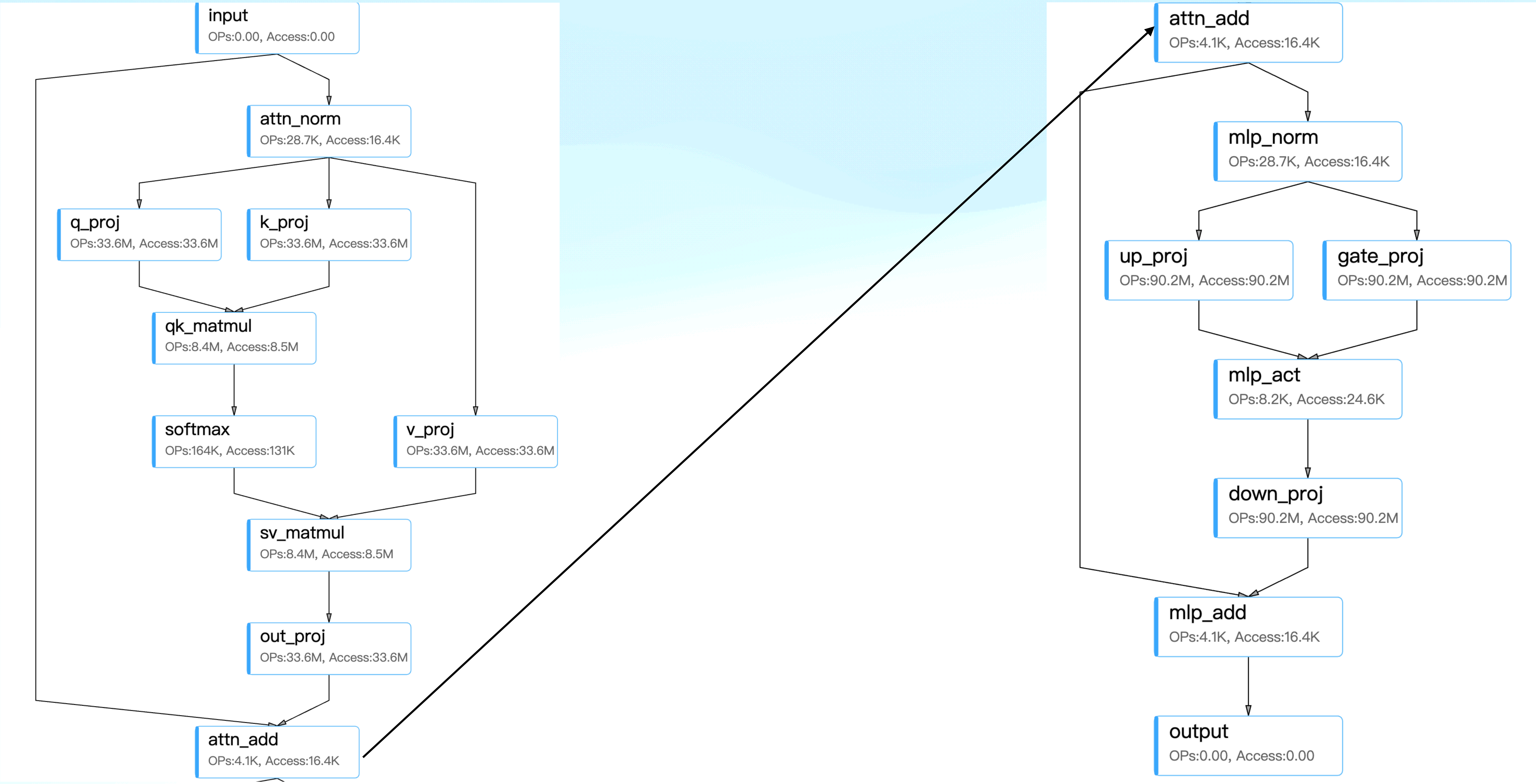
Profile result

8/23

- Profile
  - End-to-End operation analysis
  - Kernel implemented by Flashinfer
- Fuse all kernel to one

End-to-End Analysis

Transformer layer (Llama) operation



input
OPs:0.00, Access:0.00

attn_norm
OPs:28.7K, Access:16.4K

q_proj
OPs:33.6M, Access:33.6M

k_proj
OPs:33.6M, Access:33.6M

qk_matmul
OPs:8.4M, Access:8.5M

softmax
OPs:164K, Access:131K

v_proj
OPs:33.6M, Access:33.6M

sv_matmul
OPs:8.4M, Access:8.5M

out_proj
OPs:33.6M, Access:33.6M

attn_add
OPs:4.1K, Access:16.4K

MHA

attn_add
OPs:4.1K, Access:16.4K

mlp_norm
OPs:28.7K, Access:16.4K

up_proj
OPs:90.2M, Access:90.2M

gate_proj
OPs:90.2M, Access:90.2M

mlp_act
OPs:8.2K, Access:24.6K

down_proj
OPs:90.2M, Access:90.2M

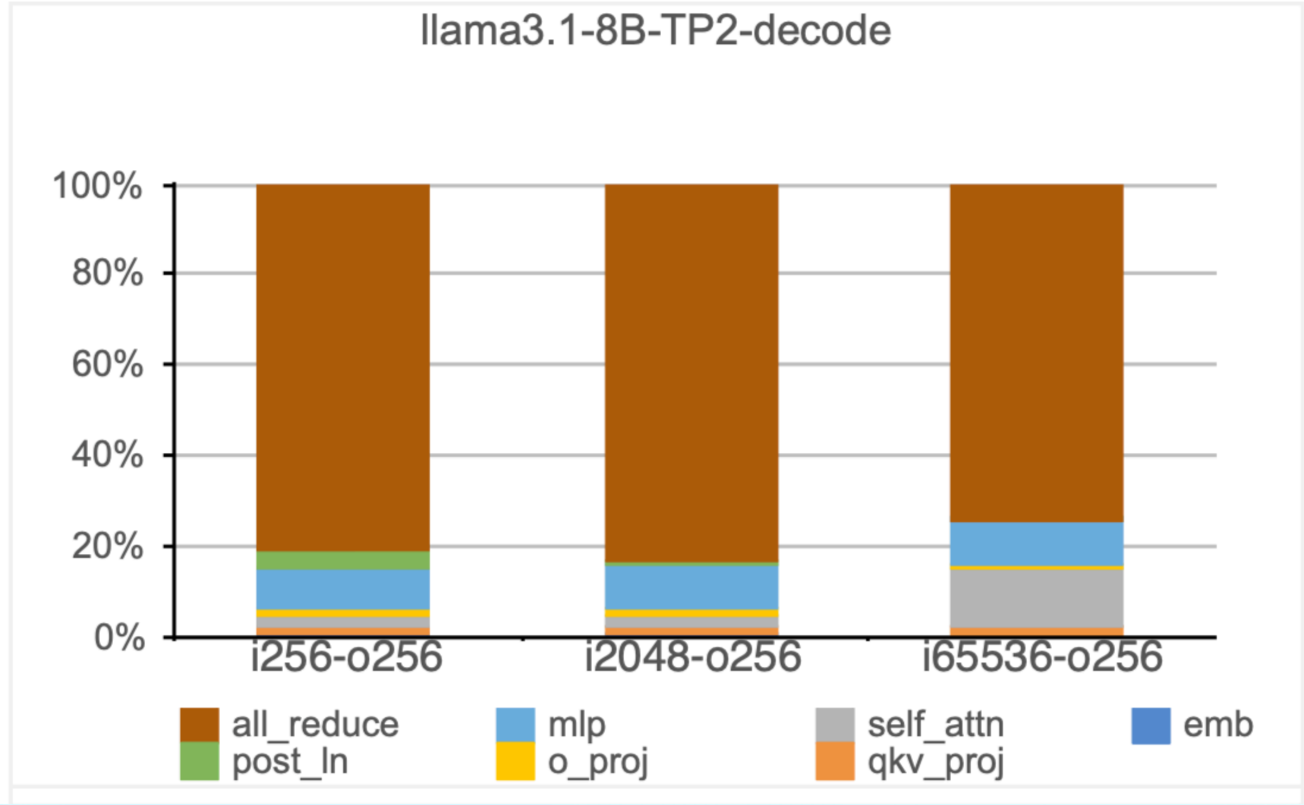
mlp_add
OPs:4.1K, Access:16.4K

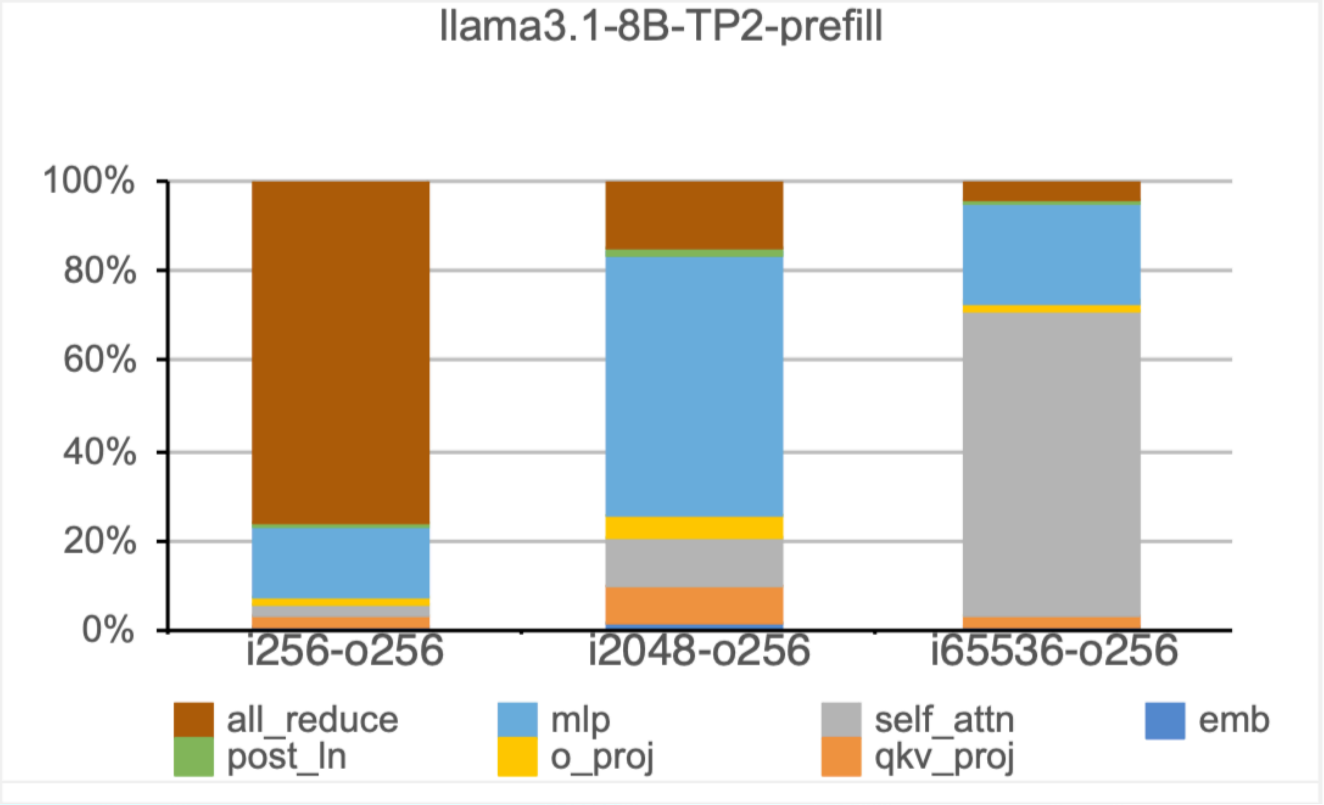
output
OPs:0.00, Access:0.00

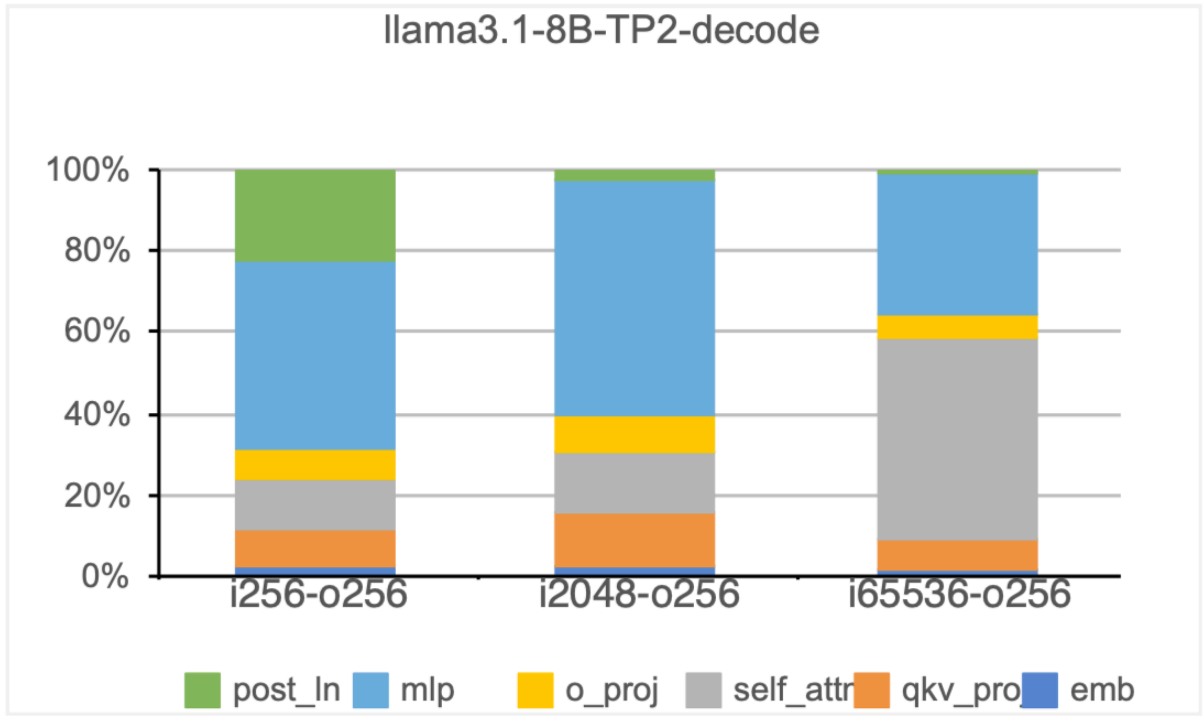
MLP

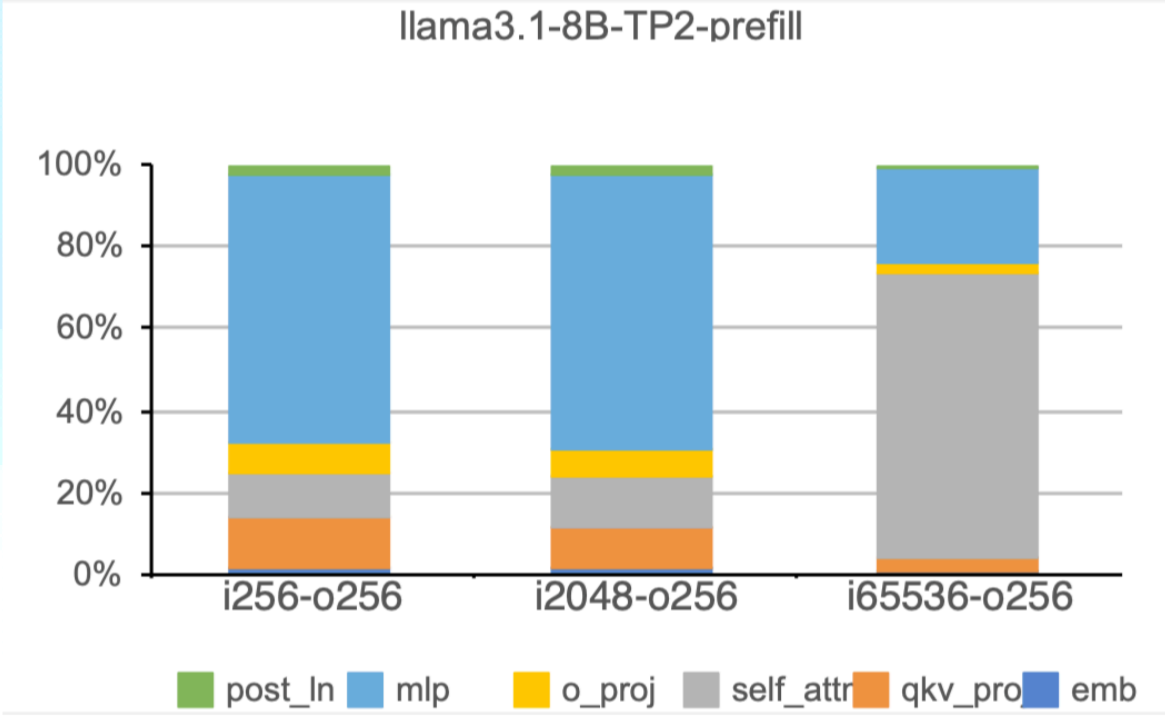
End-to-End Analysis

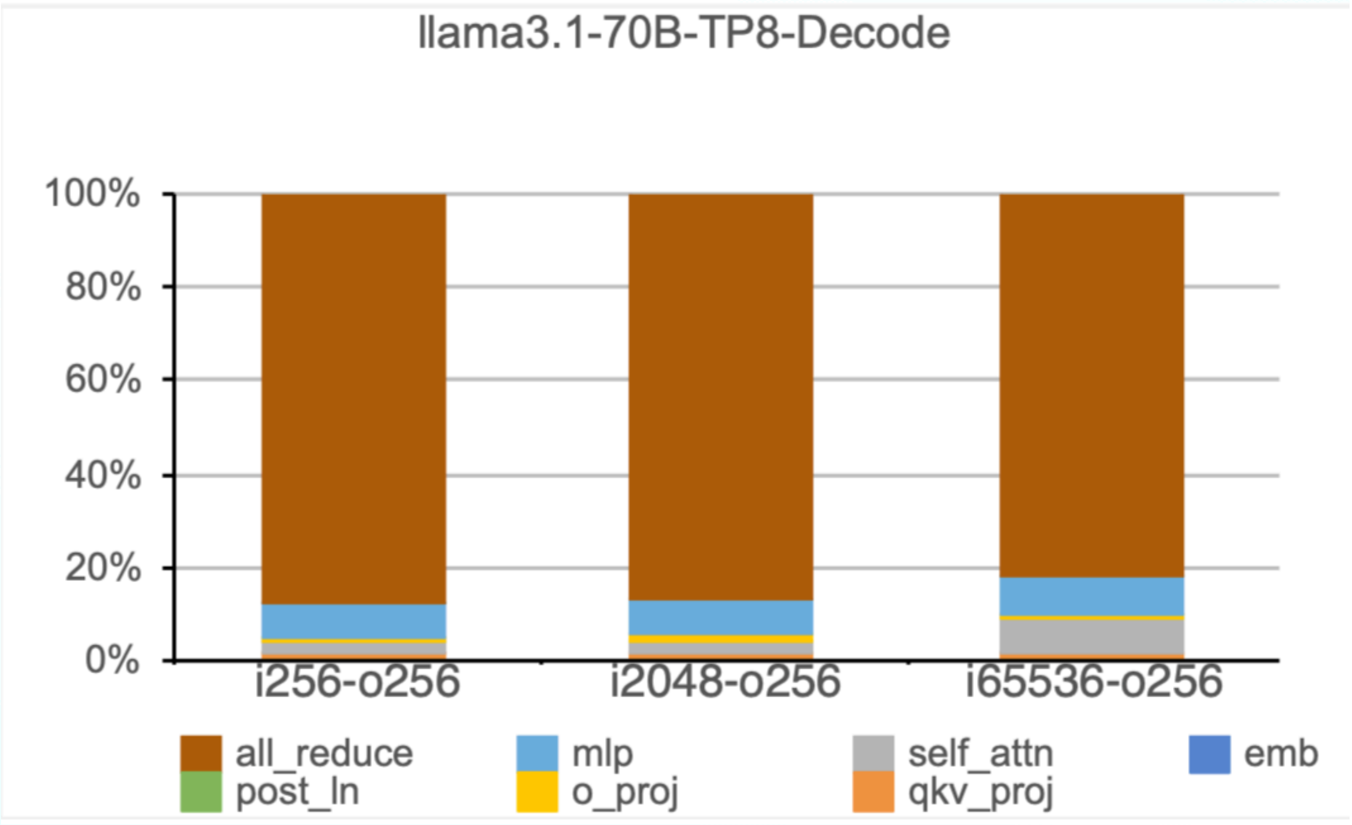
Llama3.1 operation ratio

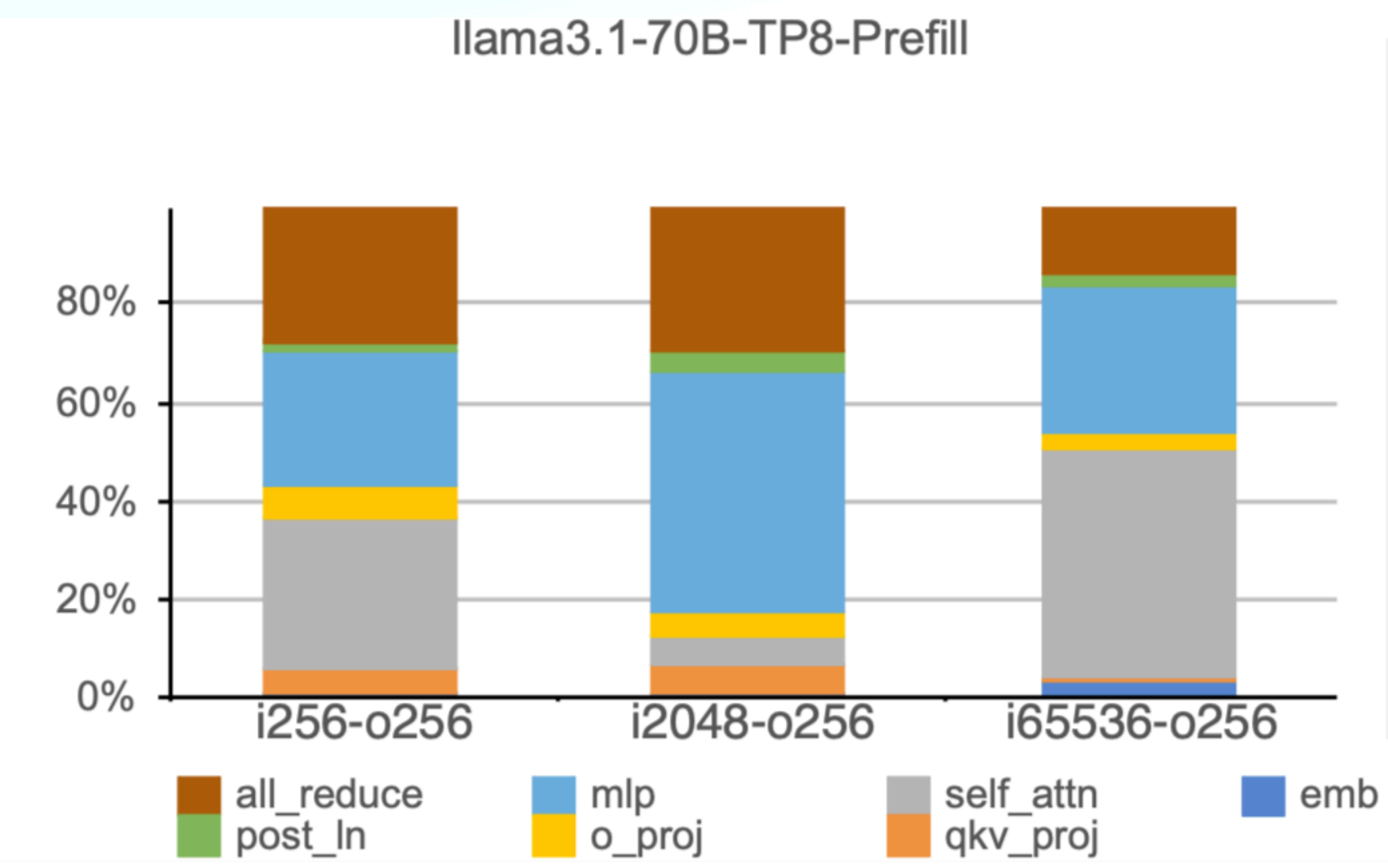


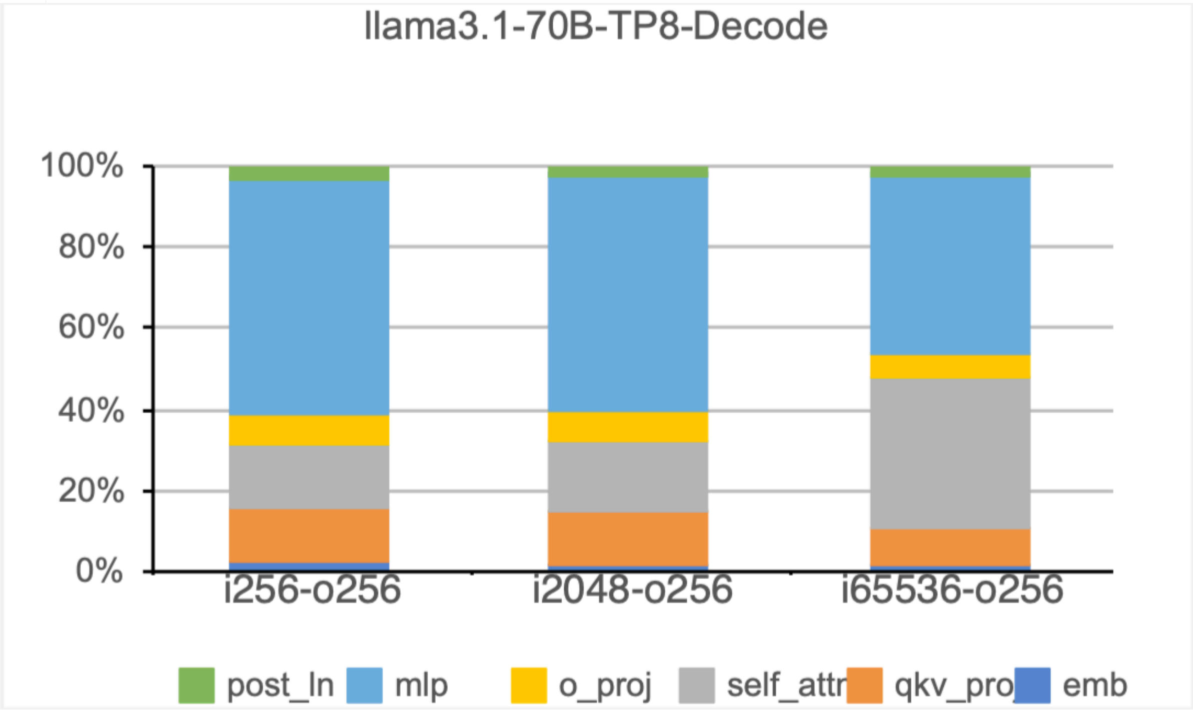


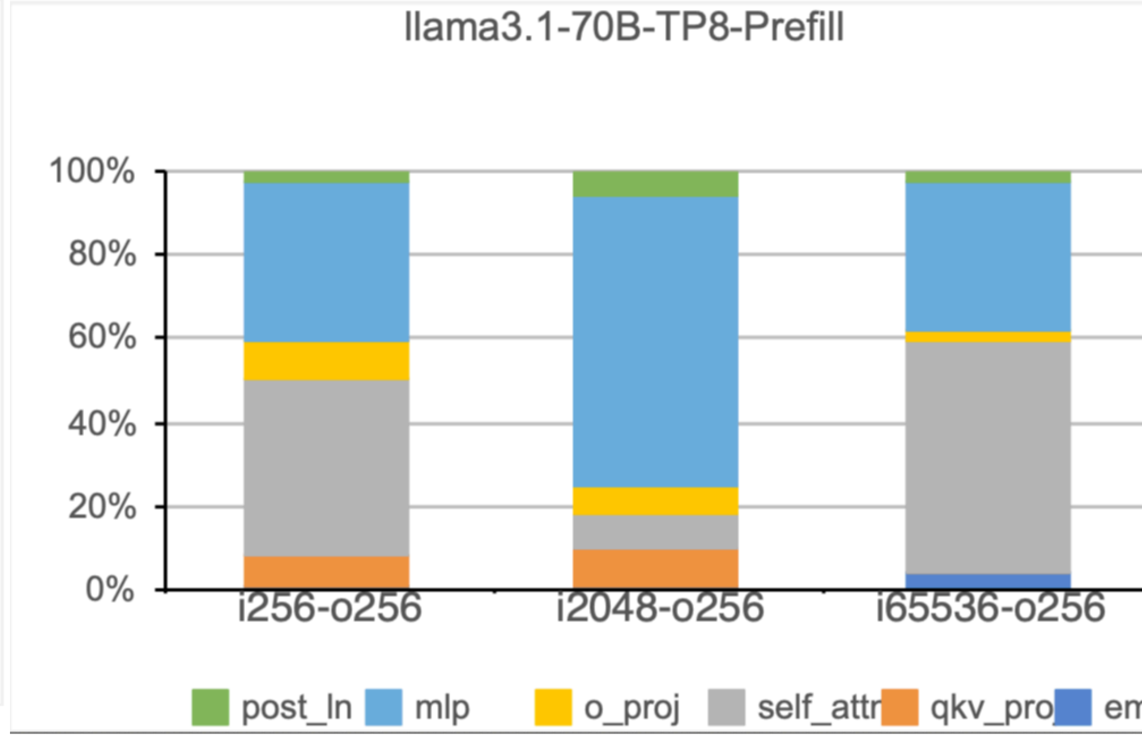






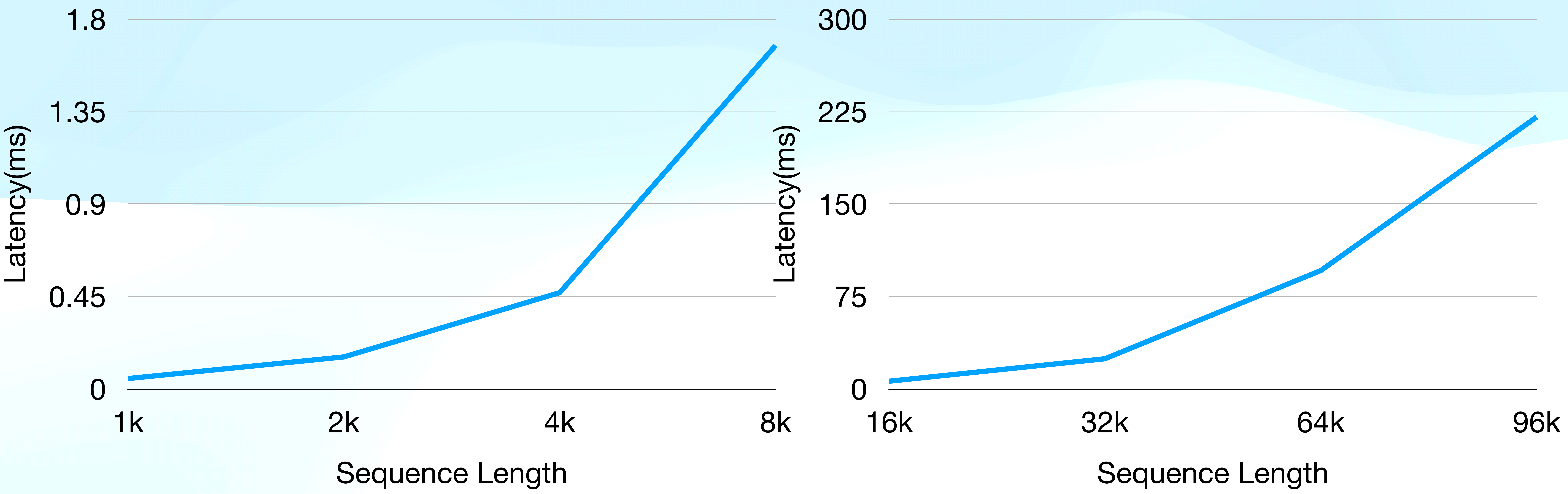






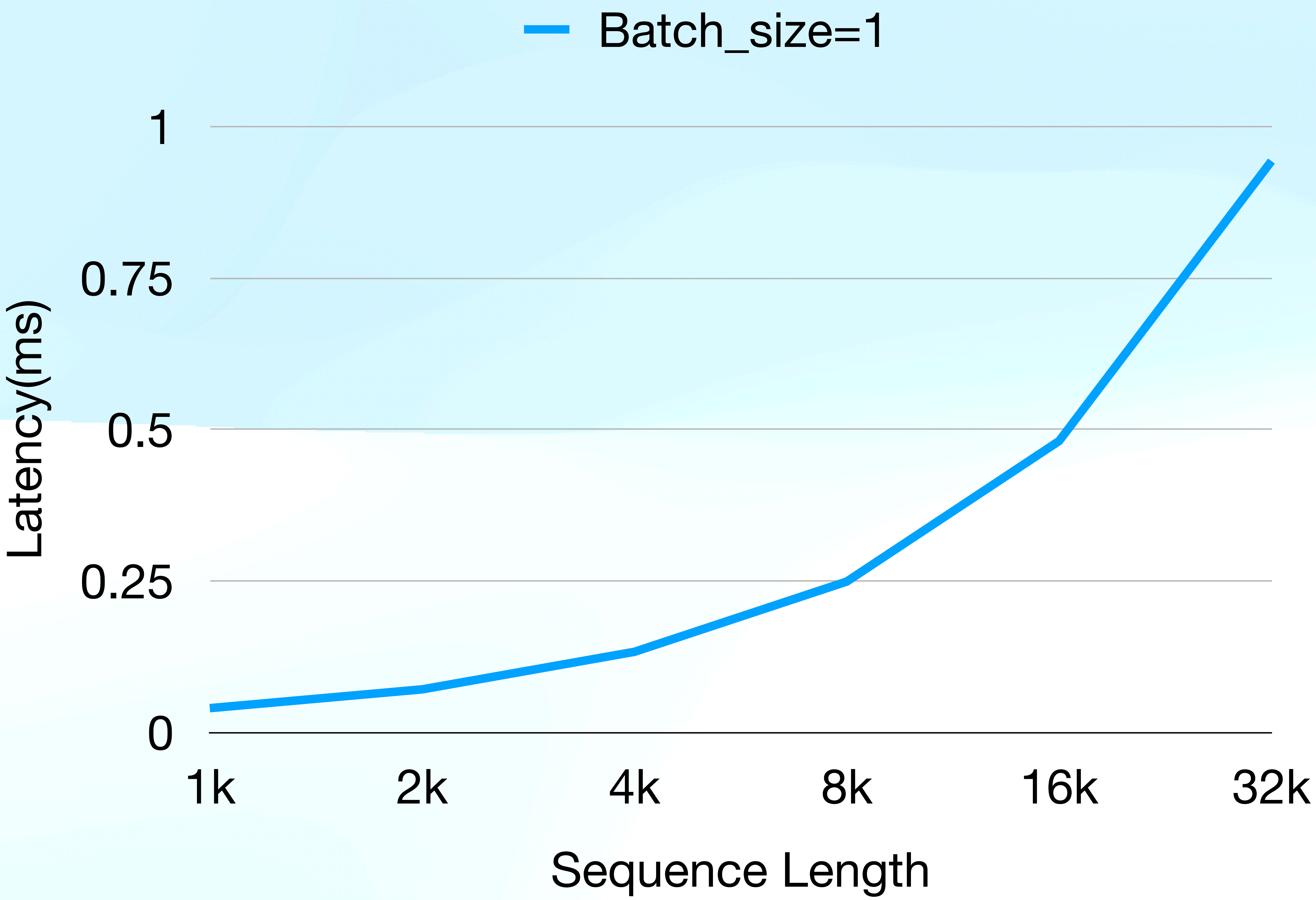
Attention Kernel Latency

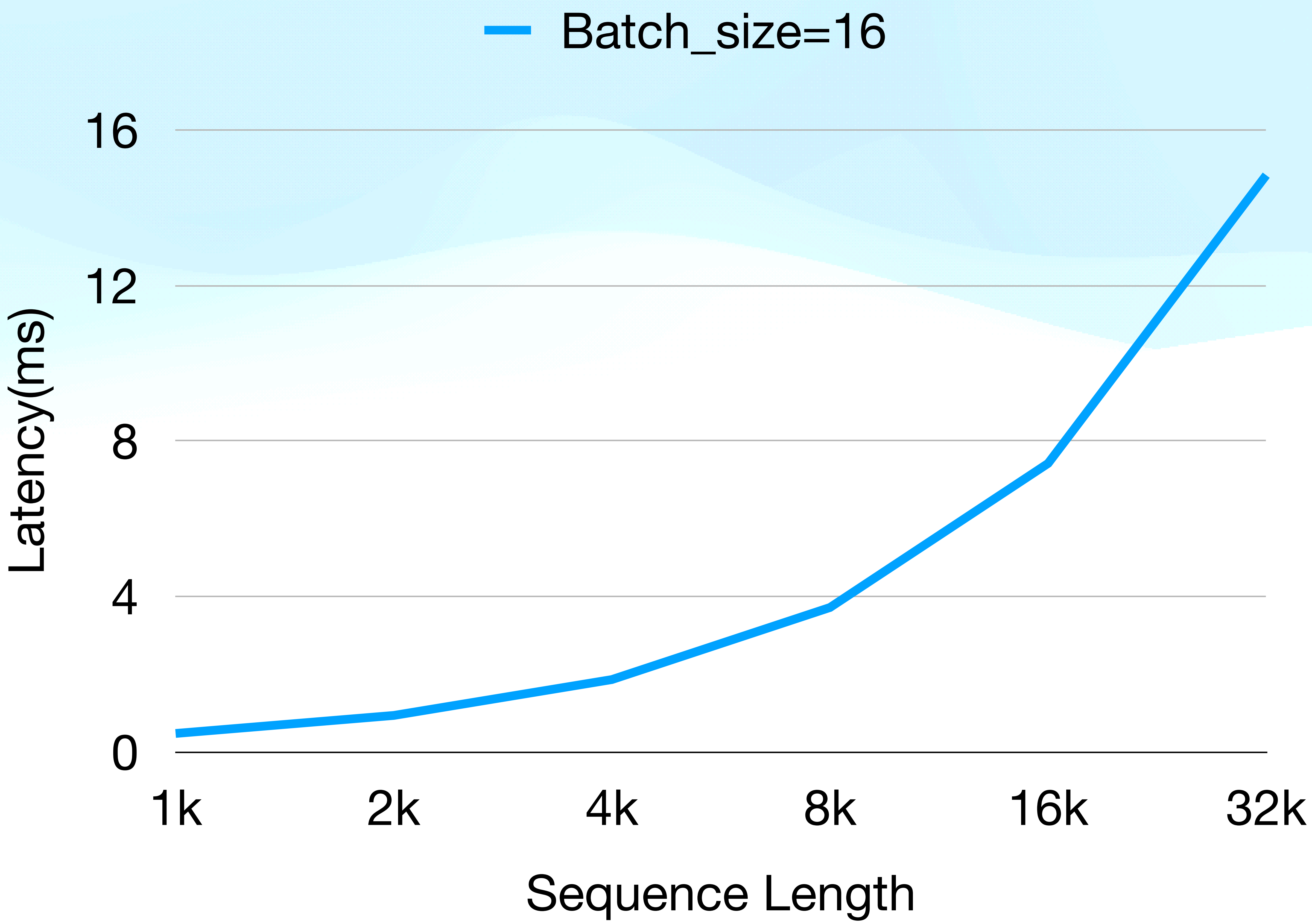
Prefill phase



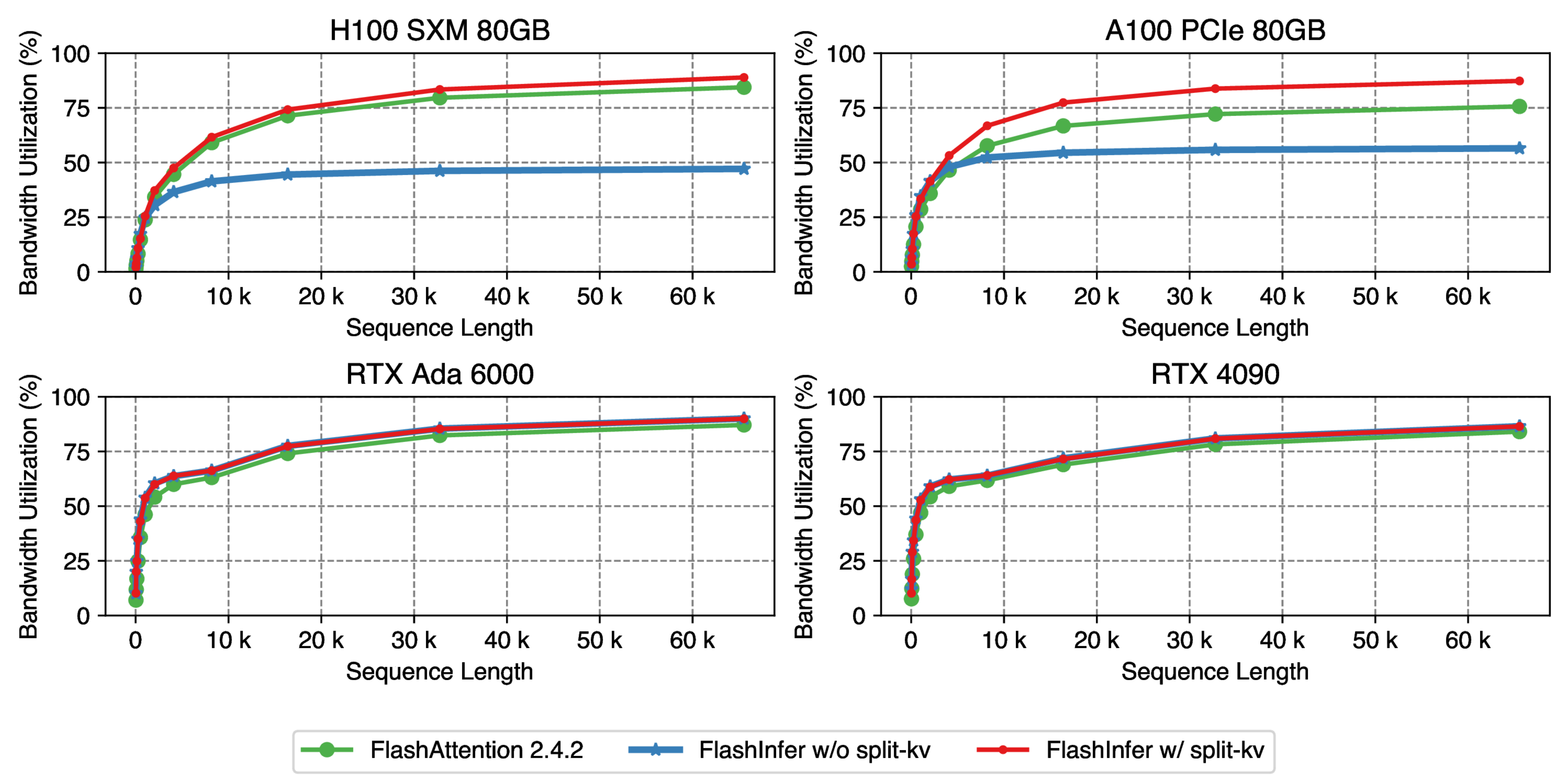
Attention Kernel Latency

Decode phase





Profile Analysis



Single request decode kernel GPU utilization

	CuBLAS	H100 Peak Tensor Cores Performance
TFlops	670	989

CuBLAS GEMM Performance

We can try to optimize through fusing all kernel to one for better end-to-end performance.

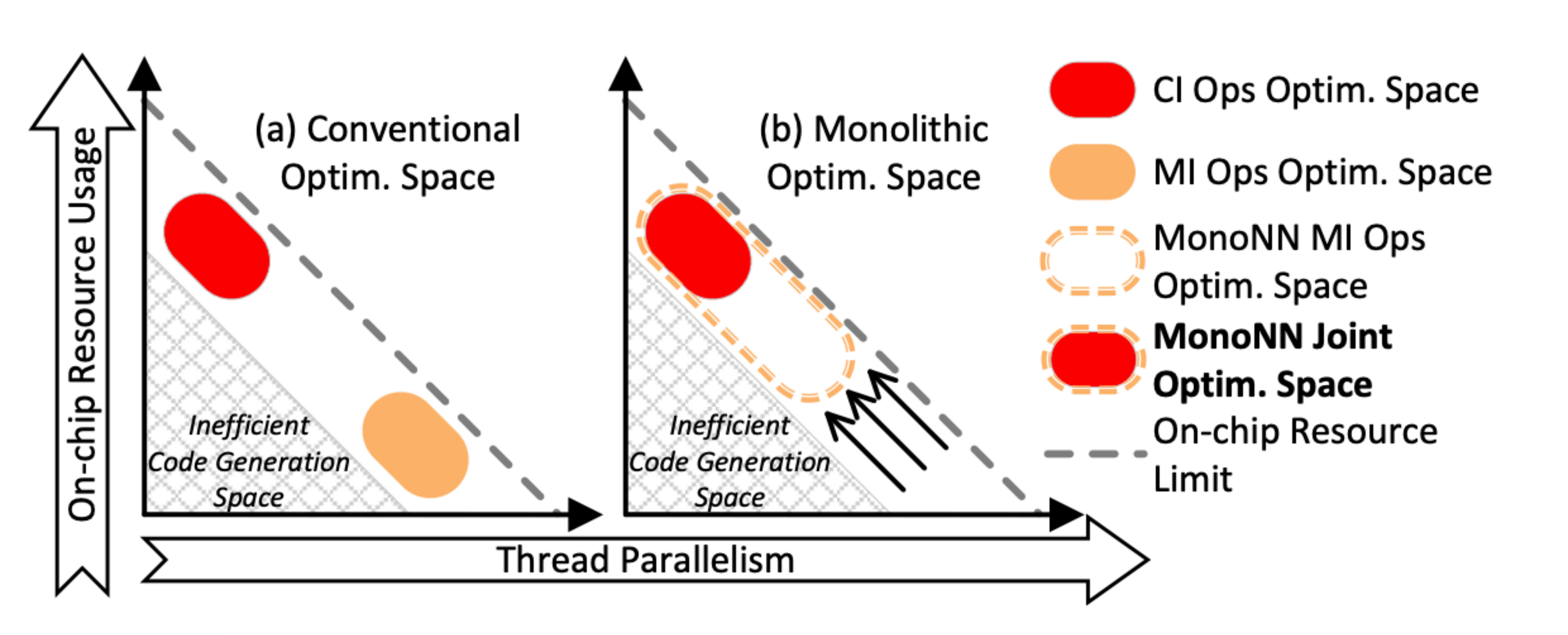
Fuse all kernel to one

Challenges

- Resource incompatibility between computeintensive and memory-intensive operators.
    - Compute-intensive operators require a large amount of on-chip resources (registers and shared memory)
    - Memory-intensive ops rely on massive concurrent threads to hide off-chip memory access(TLP).

Fuse all kernel to one

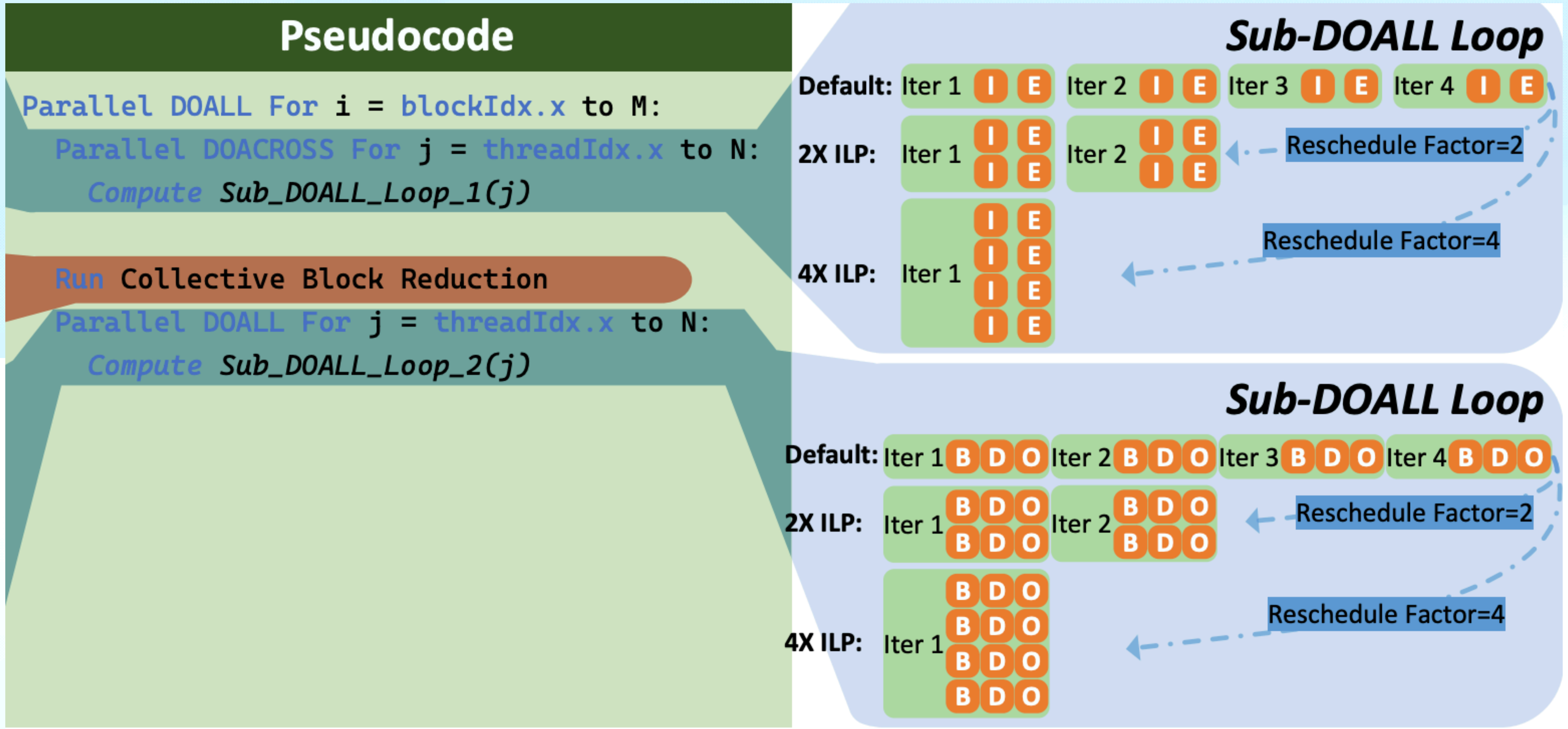
A High-level illustrate


On-chip Resource Usage
(a) Conventional Optim. Space
Inefficient Code Generation Space
(b) Monolithic Optim. Space
Inefficient Code Generation Space
Thread Parallelism
CI Ops Optim. Space
MI Ops Optim. Space
MonoNN MI Ops Optim. Space
MonoNN Joint Optim. Space
On-chip Resource Limit

Align the optimization space of MI Ops as closely as possible with that of CI Ops

Fuse all kernel to one

Detailed Implementation



Pseudocode

```
Parallel DOALL For i = blockIdx.x to M:
  Parallel DOACROSS For j = threadIdx.x to N:
    Compute Sub_DOALL_Loop_1(j)

  Run Collective Block Reduction
  Parallel DOALL For j = threadIdx.x to N:
    Compute Sub_DOALL_Loop_2(j)
```

Sub-DOALL Loop
Default: Iter 1 I E Iter 2 I E Iter 3 I E Iter 4 I E
2X ILP: Iter 1 I E I E Iter 2 I E I E
Reschedule Factor=2
4X ILP: Iter 1 I E I E I E I E
Reschedule Factor=4
Sub-DOALL Loop
Default: Iter 1 B D O Iter 2 B D O Iter 3 B D O Iter 4 B D O
2X ILP: Iter 1 B D O B D O Iter 2 B D O B D O
Reschedule Factor=2
4X ILP: Iter 1 B D O B D O B D O B D O
Reschedule Factor=4

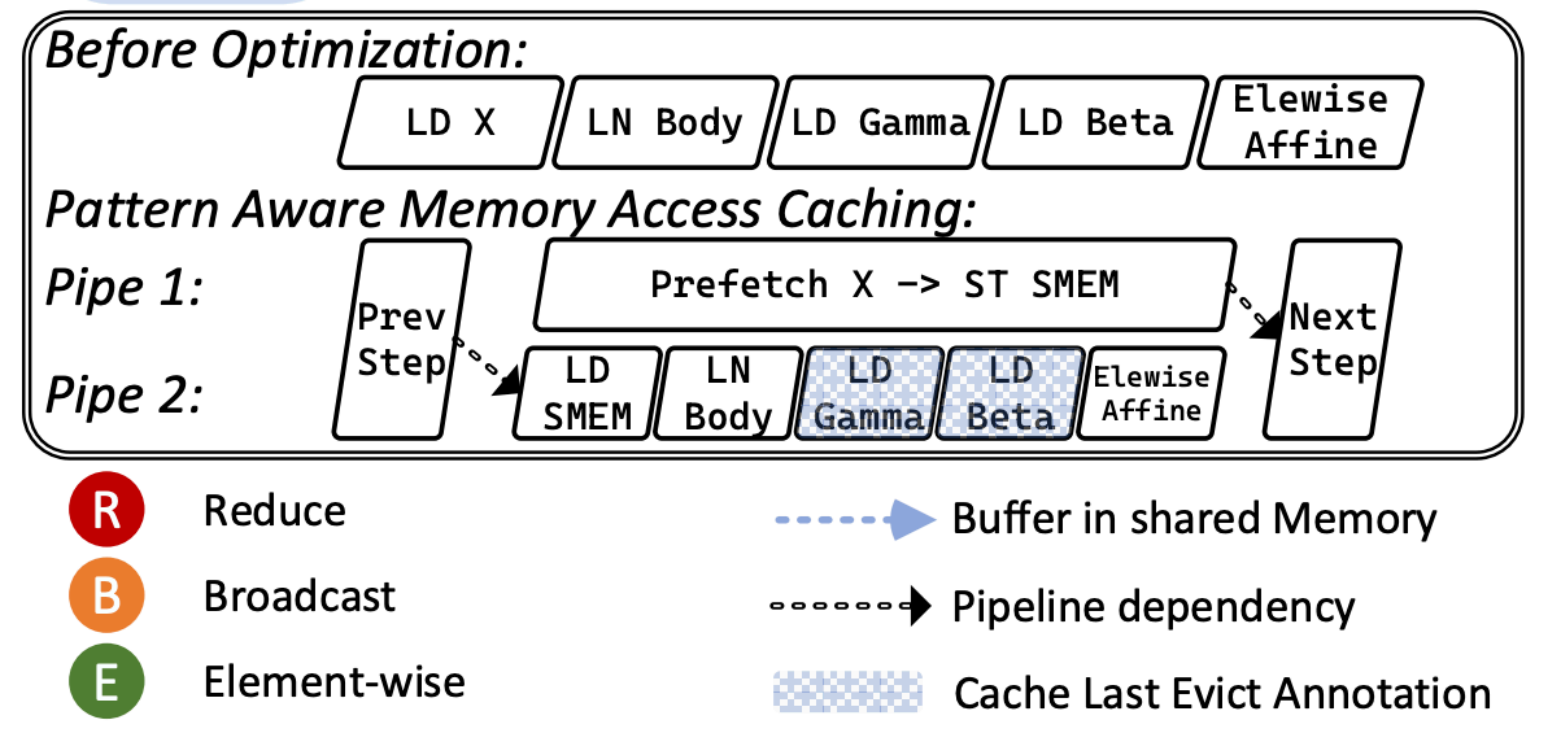
Context-aware instruction rescheduling for softmax computation.

Context-aware instruction rescheduling:

For memory-intensive operator(softmax), we can use ILP(Loop unroll) which need more on-chip resources adaptive with compute-intensive operator(GEMM).

Fuse all kernel to one

Detailed Implementation



On-chip resource exploitation

Memory access optimizations:

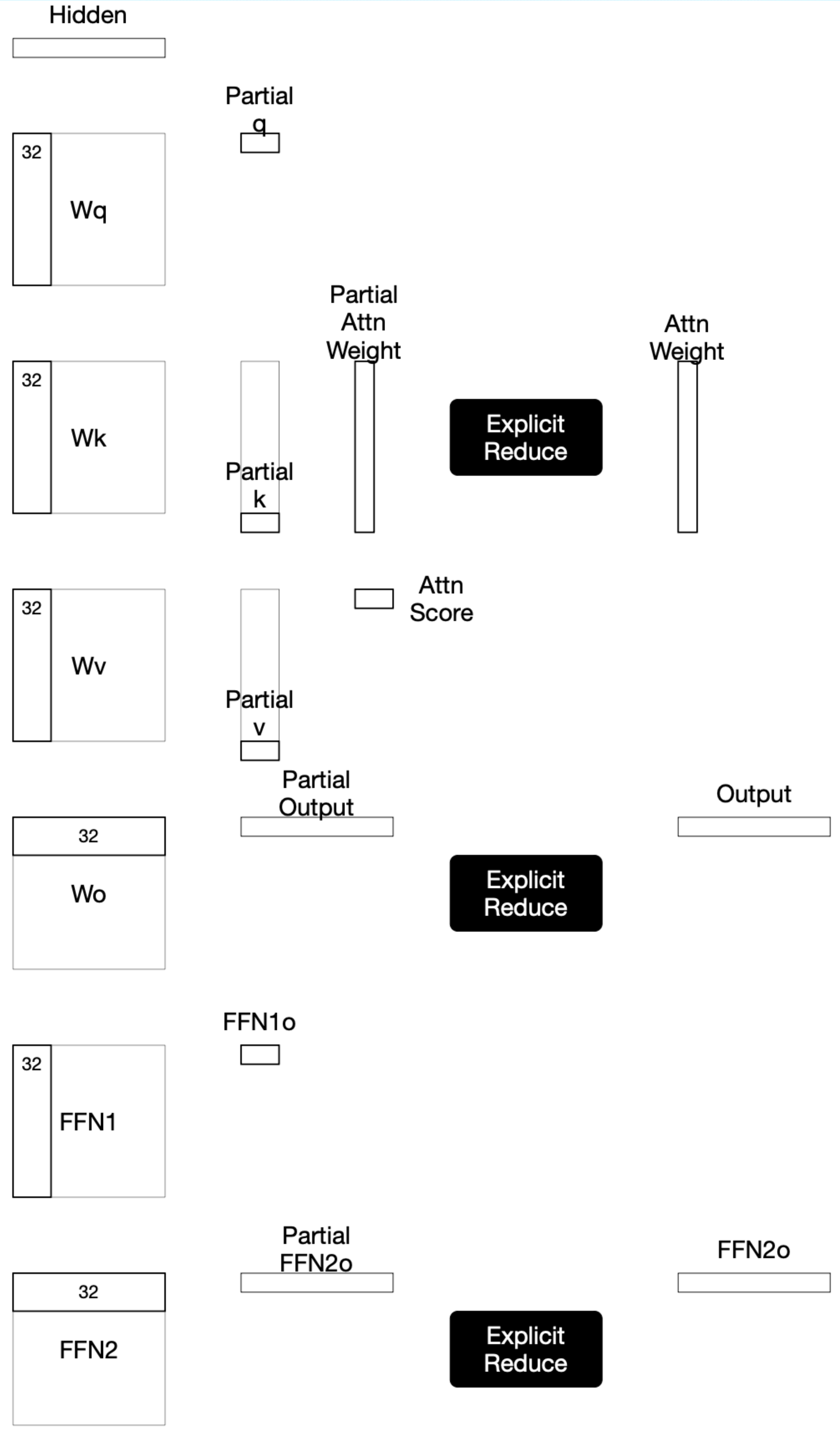
Leverages the abundant shared memory resource

allocated by the compute-intensive computation

in one kernel to pipeline the global memory access

Fuse all kernel to one

Fuse decode phase to one



Thanks